AQM's for achieving fairness between competing TCP flows

Rade Stanojević and Robert Shorten

***Abstract*— Making drop decisions to enforce max-min fair resource allocation without any explicit information is a challenging problem. Here we develop a spectrum of stateless queue management schemes: MLC(l) (Multi-Level Comparison with index l). We show analytically that for arbitrary network topology of TCP users and queues employing MLC(l) resource allocation converge to max-min fair, as l grows. Our second algorithm MAY (Markov Active Yield) exploits the following surprising observation: the information on dropped packets is enough for enforcing max-min fairness among arbitrary set of AIMD flows. MAY enforce max-min fairness by keeping only the information on short history of already dropped packets. By doing this amount of memory required for storing the state can be significantly reduced. The properties of MLC(l) and MAY are evaluated and characterized theoretically using a Markov chain model, and experimentally using packet level ns2 simulations.**

***Index Terms*— AQM, congestion control, max-min fairness, Markov chains, TCP.**

I. Introduction

Resource allocation has been an important research problem in communication networks for more than a decade[25]. In the current Internet most of traffic uses TCP as transport protocol, and most of Internet routers are oblivious: they do not differentiate packets from different flows. In order to change resource allocation, one can do the following: (1) Design the new end-to-end protocol(s) and leave the network infrastructure (routers) unchanged[15], [17], [16]. (2) Design the new end-to-end protocol(s) and network support that will allow cooperation between end-users and network; (routers)[13], [1], [14]; (3) Leave the end-to-end protocol(s) unchanged but design the network based scheme that determines desired resource allocation[10], [7], [8].

Most of current proposals have as a performance objective resource allocation that is max-min fair. In this paper we propose two schemes MLC and MAY that belong to the third group whose performance goal is enforcing max-min fair resource allocation. The main features are:

- They do not require change in the end-to-end transport protocol TCP.

- Since end users use TCP, the only congestion indicator is binary: packet drop or ECN marking.

- Decision which packet to drop (mark) is made by each router independently with locally available information.

- No multiple queues neither per flow counters are used.

Briefly, our goal is:

The design of a stateless active queue management scheme that can enforce max-min fairness in the network of TCP users.

While there exist a large amount of work related to analysis and design of distributed algorithms that enforce max-min fairness to the best of our knowledge this is the first attempt to design a stateless active queue management scheme that enforce max-min fairness in the network of TCP users.

A. Paper contributions

Why reaching the goal stated above is hard? First, recall that in the max-min fair regime, a TCP flow f experiences drops at *one and only one* link l_f at its path (we say that f is bottlenecked at l_f), and therefore must be protected at other links (that can be congested) by receiving lossless service. Second, if two or more flows are bottlenecked at the same link they must receive nonuniform loss rates that are function of their aggressiveness (Round-trip times, queuing delays, etc). Assuming the knowledge on flow rates (written in IP header[10] or estimated by the router itself[11]) or existence of multiple queues that are appropriately scheduled a number of solutions to this two problems exist and are developed in previous work. However in our case it is highly nontrivial to make the drop (mark) decision without any explicit information.

The main contributions of this paper are:

- The stateless queue management scheme Multi Level Comparisons with index l (MLC(l)) which makes the drop decision based on the structure of packets that are already in the queue.

- The queue management scheme Markov Active Yield: MAY. The *only* state information that is kept by MAY is a short history of *already dropped packets*.

- The Markov chain analysis of the randomized algorithms MLC(l) and MAY that show that (1) Resource allocation of MLC(l) converge to the max-min fair, for arbitrary network topology and arbitrary set of TCP users, as index l grows. (2) The resource allocation of TCP users using MAY queue at single bottleneck topology is max-min fair (each TCP users receive same throughput).

- The packet level simulations that supports analytical findings and suggest that MAY resource allocation is very close to max-min fair for general network topologies.

As we will see later the router based algorithm MLC(l) can be seen as a dual to well known Mo&Walrand[23] end-to-end (p, α)-proportionally fair algorithm. While MAY algorithm is not completely stateless it only keep state of flows that have experienced drops which can be significantly smaller than keeping per flow state at heavy-tailed flow size/rate environments which are typical at Internet links. Quite surprisingly, we show that information on dropped packets is enough to establish an dropping decision algorithm which allocates bandwidth fairly amongst TCP users.

B. Related work

Allocating bandwidth fairly amongst competing users in the Internet is an important task. The central point in the current Internet is the fact that most of traffic is generated by responsive TCP users. However, the level of responsiveness of an individual user might depend on implementation of transport protocol as well as global characteristics of flow such as round trip time or queueing delays. Schemes to ensure fair bandwidth allocation would play a role of protecting less aggressive users from aggressive ones and therefore would allow existence of various end-to-end congestion control algorithms. Motivated by the widespread deployment of highspeed links, many new aggressive congestion control protocols have been recently proposed [17], [16]. Such protocols are known to be very aggressive when competing with standard TCP, and in such environments, it is likely that some form of forced fairness is necessary to protect less aggressive flows. Thus, if we assume networks with heterogenous end-to-end congestion control strategies, the only way to control fairness in the network is to do it at routers using some form of queue management.

The design of router queue management algorithms has been a very active research area during last decade. Typically, a number of separate strategies can be discerned when reviewing this work; those strategies that use a single queue and do not require the router to process state information; those that require the router to process some state information and/or may utilize multiple router queues; and those that require changes to existing IP infrastructure. We use the term (AQM) to refer to the first of these, Fair Scheduling to refer to the second of these, and New Architecture Proposals to refer to the last of these.

(1) *Active Queue Management(AQM) :* Traditional AQM's [3], [4], [5], [6] have been designed with the ultimate goal of providing high link utilization, low queueing delays, and small number of packet losses. A key issue in the design of AQM schemes is the complexity of their implementation and the requirement that they should be implementable on a wide range of Internet routers. As most AQM schemes do not differentiate packets from different flows, bandwidth allocation among users is mainly determined by end-to-end congestion control policies. For example, over routers with constant loss probability, the amount of bandwidth that standard TCP user obtain is proportional to inverse of its round-trip time[29], [28].

(1a) *CHOKe :* CHOKe[27] is an example of an AQM that differentiates packets from different flows; flows with higher bandwidth are punished more than in RED or BLUE for example. This has the effect of improving fairness amongst responsive users, and the degree of improvement for some AQM schemes based on the basic CHOKe idea will be examined in Section II

(2) *Fair Scheduling:* Fair scheduling mechanisms[7], [8], [9], [11], [12], require the router to maintain some per-flow state information and to use this information to manage arriving packets. Although they achieve excellent fairness, they are not widely deployed in real networks, mainly because high computational costs associated with obtaining, processing and managing this state information.

(3) *New Architecture Proposals :* Several new proposals, including CSFQ[10], XCP[13] and MaxNet[14], require changes to the packet IP header. Although they perform quite well in terms of fairness and resource utilization, they assume cooperation of most (if not all) routers in the entire network which is clearly very restrictive.

II. CHOKE-LIKE AQM SCHEMES

It has been noticed in many studies that both drop tail and RED routers have large bias against large-RTT flows. For example Lakshman&Madhow [19] have made the empirical observation that for a drop-tail router and two flows with round trip times RTT_1 and RTT_2, the ratio of asymptotic throughput of the first and the second flow is in the ratio $(RTT_2/RTT_1)^a$ for some $a \in (1, 2)$. Similarly, it has also been noticed in a number of studies, that oblivious (ones that do not differentiate packets from different flows) AQM schemes (RED [3], BLUE [4], etc.) which attempt to estimate the loss probability for a given traffic pattern and to drop packets according to this estimation, share bandwidth among competing users with round trip times RTT_1 and RTT_2 in the ratio RTT_2/RTT_1, [29], [28]. In this section we will investigate RTT unfairness characteristics for more general AQM schemes. In particular, we shall characterize the fairness properties of CHOKe-like AQM schemes.

Definition 2.1: An AQM is CHOKe-like if it drops a packet from a flow with current throughput[1] U with probability $\rho_0 U^{l-1}$, where ρ_0 is variable controlled by router and l positive integer called index of given CHOKe-like AQM.

Comment : With $l = 1$ this corresponds to a router which drops packets with loss probability ρ_0, exactly as BLUE in steady state. The case when $l = 2$ is similar to CHOKe in the limit when the average queue size does not go the below minimum threshold, and in addition, when there is neither a RED nor an overflow drop. Indeed, comparing a packet at the entrance of queue with a packet from the queue and making drop-decision based on this comparison is actually dropping a packet with probability which is proportional to current throughput of the flow.

In this section we will describe a class of CHOKe-like AQM's called Multi Level Comparison (MLC) AQM's. In particular, we will describe and analyze the fairness characteristics of this queueing discipline for TCP flows competing for bandwidth.

We will see that the MLC scheme with index l achieves $1/RTT^{1/(l+1)}$-fairness[2] under the assumption of low loss

[1]Throughput is measured in packets per unit of time.

[2]Two flows with round trip times RTT_1 and RTT_2 which have a single bottleneck operating with MLC, obtain bandwidth in ratio: $(RTT_1/RTT_2)^{1/(l+1)}$.

probability. More generally, Theorem 2.2 shows that increasing l leads to fairness among TCP users arbitrary close to max-min in general network topologies.

A. Description of MLC

The basic strategy in MLC(l) is to extend the core idea from CHOKe of comparing of a packet arriving at the queue with packets which are already in queue; these stored packets are a measure of the proportion of bandwidth used by certain flow. MLC(l) maintains a variable h_M which is used to control the probability of dropping an arriving packet: at every packet arrival h_M dropping trials are executed.

Dropping trial: Pick randomly $l-1$ packets from the queue: if *all* l packets belong to same flow, then the arriving packet is dropped[3].

If the arriving packet is not dropped after execution of h_M dropping trials then it is enqueued. If h_M is not an integer, the number of dropping trials is given as follows. For $h_M < 1$ we execute 1 dropping trial with probability h_M and 0 dropping trial with probability $1-h_M$. Similarly, $h_M > 1$ we execute $\lfloor h_M \rfloor + 1$ dropping trials with probability $\{h_M\} = h_M - \lfloor h_M \rfloor$ and $\lfloor h_M \rfloor$ dropping trials with probability $1 - \{h_M\}$.

Proposition 2.1: For a given h_M, MLC(l) is CHOKe-like scheme with index l.

Proof: Let U_f be the throughput of a flow f, and U_0 the aggregate throughout on the link. A packet is dropped at one dropping trial with probability

$$q_1 = \left(\frac{U_f}{U_0}\right)^{l-1}.$$

Probability that a packet is dropped after h_M trials is 1-p[packet is not dropped at any of h_M trials] which is given by

$$q = 1 - (1-q_1)^{h_M} \approx q_1 \cdot h_M = U_f^{l-1} \frac{h_M}{U_0^{l-1}}.$$

Taking $\rho_0 = \frac{h_M}{U_0^{l-1}}$ we conclude that MLC(l) satisfies Definition 2.1. ■

The higher h_M the more frequent the losses are. Consequently, if the link is under-utilized h_M should be decreased in order to decrease probability of dropping packets. On the other hand if the aggregate traffic on the link is grater then the link capacity then h_M should be increased to reduce traffic load.

Controlling the variable h_M : MLC uses a parameter $SampleTime$ to affect changes in the variable h_M (in our simulations this is set to $100ms$). h_M is adjusted once per $SampleTime$ using a MIMD (Multiplicative Increase - Multiplicative Decrease) scheme; see Figure 1. The performance goal is to keep the utilization at a certain level u_0. Namely, if within the previous $SampleTtime$ the link utilization was less than desired u_0, h_M is set to γh_M for some $\gamma \in (0,1)$, otherwise h_M is adjusted as $h_M := h_M \delta$ for some $\delta > 1$.

[3]If $l = 1$ the arriving packet is dropped by default.

```
UpdateFrequency($h_M$)
  if ($now - LastUpdate > SampleTime$)
    if $utilization < u_0$
      $h_M = h_M \cdot \gamma$;
    else
      $h_M = h_M \cdot \delta$;
    endif
    $LastUpdate = now$;
  endif;
```

Fig. 1. Control of h_M

At this point it is important to emphasize a few differences between MLC and CHOKe. First, note that CHOKe makes a comparison only when the average queue size becomes greater than min_{th} (RED minimum threshold), and therefore its performance (in terms of resource allocation between TCP users) depends mainly on the number of users: a small number of users will affect the synchronization of losses, while for large number of users, the number of CHOKe-drops will be much less then number of RED-drops and therefore the effect of CHOKe to TCP fairness would be negligible. Second, in MLC, if a packet is not dropped at the entrance of the queue, it will not be dropped latter while in the queue. On the other hand the original CHOKe algorithm allows dropping both arriving packet and packet(s) from the queue, for the purpose of reducing the unresponsive flow(s) throughput.

Our experiments indicate that the parameter $SampleTime$ should be in the range of round trip times of the connections using the link (in order to allow users to react to changes in h_M). The parameters γ and δ control the speed of adaptation to changes in network traffic and are set such that, for a given $SampleTime$, h_M can be doubled/halved within a few seconds.

B. Model and analysis of CHOKe-like AQM

In this section we present a model of CL-AQM's servicing multiple TCP users. We present results that characterize this situation for both a single bottleneck and for general network topologies.

Single bottleneck case: We consider N TCP-flows with heterogenous round trip times RTT_i, $i = 1, \dots, N$, traversing a single bottleneck link that employs CL-AQM with an index l. If we assume that ρ_0 does not fluctuate much so that we can model it as constant and that the drop probability for a packet is small, then our analysis shows that the asymptotic rates achieved by TCP users are proportional to $\frac{1}{RTT_i^{2/(l+1)}}$. This is the main result of this section and is given in Theorem 2.1.

Model : At the flow level, let Δ to be length of sampling interval over which we evaluate changes in throughput. If a flow with a round trip time RTT does not see a drop within interval of length Δ, then its throughput will be increased for Δ/RTT^2. If a flow registers a drop within this sampling interval then its throughput will be halved[4]. The probability that the first event will happen is equal to probability that each of ΔU packets from the flow are not dropped. This probability

[4]Throughout this paper, variations in round trip times are neglected.

is given by:

$$\eta_1 = (1-\rho_0 U^{l-1})^{\Delta U} \approx e^{-\Delta\rho_0 U^l}.$$

Clearly, the probability that a flow with current throughput U will see a drop within a sampling interval of length Δ is equal to

$$\eta_2 = 1-(1-\rho_0 U^{l-1})^{\Delta U} \approx 1-e^{-\Delta\rho_0 U^l}.$$

The previous approximations are valid under the assumption of a small probability that a packet will be dropped : $\rho_0 U^{l-1} \ll 1$. This assumption seems reasonable, since if this probability is not small, a flow would suffer too many losses an therefore would not get chance to grow.

Let $U_k^{(\rho)}$ be a stochastic process which describes the evolution of throughput of a TCP flow with round-trip time RTT traversing over link with a CHOKe-like AQM scheme with index l. Here $\rho = \Delta\rho_0$. Since Δ is fixed we can assume that Δ to be equal to one unit of time.

We model $U_k^{(\rho)}$ as a Markov chain on $[0,\infty)$ defined by $U_0^{(\rho)} = 0$ and:

$$U_{k+1}^{(\rho)} = U_k^{(\rho)} + \frac{1}{RTT^2} \quad \text{with probability} \quad e^{-\rho(U_k^{(\rho)})^l}$$

$$U_{k+1}^{(\rho)} = \frac{1}{2}U_k^{(\rho)} \quad \text{with probability} \quad 1-e^{-\rho(U_k^{(\rho)})^l}.$$

The following theorem characterizes the time averaged throughput of a TCP flow with round trip time given by RTT, running over CL queue management scheme with index l.

Theorem 2.1: The time averaged throughput of the i'th flow: $\frac{1}{M}\sum_{i=1}^{M} U_i^{(\rho)}$ converges almost surely to:

$$\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M} U_i^{(\rho)} =: \overline{U}^{(\rho)} = \frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}} D_{CL}(l) + \frac{1}{\rho^{\frac{1}{l+1}}} S^{(\rho)}$$

where $D_{CL}(l)$ is a constant that does not depend on ρ neither RTT and $S^{(\rho)}$ converges to 0 as ρ goes to 0.

Remark: The previous theorem is generalization of well known square root formula. Indeed, for $l=1$, CL scheme is a oblivious AQM that drops packets with probability $\rho = h_M$ and Theorem 2.1 says that time averaged throughput converge to $\frac{1}{RTT\sqrt{\rho}} D_{CL}(1) + o(\frac{1}{\sqrt{\rho}})$

To conclude this section we prove that for a given network with routers employing a CL AQM with index l, and assuming that the steady state throughput is given by the previous theorem, we can find large enough l such that bandwidth allocation is arbitrary close to max-min fairness. The following characterization of max-min fairness can be found in [25].

Lemma 2.1: A set of rates x_r is max-min fair if and only if for every flow r there exists a link on its path, such that the rates of all flows which traverse through that link are less or equal then x_r.

With this characterization of max-min fair allocation in mind, we shall prove that increasing the index of the CL scheme will result in allocation of bandwidth in such fashion that each flow will have link on its path such that its asymptotic rate is "almost" the largest among all flows using that link.

Theorem 2.2: For any given network topology, and given $\varepsilon > 0$, there exists l such that if all queues employ CL-AQM with index l and loss probabilities are small then for every flow r there exist a link on its path, such that the rates of all flows which traverse through that link are less then $(1+\varepsilon)x_r$ (here x_r is steady state rate of flow r).

Proof: Let L be the number of links in the network and N the number of flows. We label flows by $i = 1,2,\ldots,N$ and links by $s = 1,2,\ldots,L$. By R we denote the routing matrix: $R_{is} = 1$ if flow i uses link s otherwise $R_{is} = 0$. On each link s, a router drops a packet from the flow with current throughput U with probability $\rho(s)U^{l-1}$. Let M be the length (in number of links) of the path of the flow with most links on its route and ν the ratio of the largest and the smallest round trip time in the network. Choose l such that

$$\nu^{\frac{2}{l+1}} M^{\frac{1}{l+1}} < 1+\varepsilon.$$

For each flow r, let $s_1^{(r)},\ldots,s_w^{(r)}$ be links used by it and let $s_{max}^{(r)}$ the most congested link on its route in the following sense:

$$\rho(s_{max}^{(r)}) = \max\{\rho(s_j^{(r)}) |\ j = 1,\ldots,w\}. \tag{1}$$

If the current rate of flow r is U, a packet from that flow will be dropped with probability $\lambda_r U^{l-1}$, where $\lambda_r = \sum_{j=1}^{w}\rho(s_j^{(r)})$, and therefore the steady state throughput for flow r is given by

$$x_r = \frac{1}{RTT_r^{\frac{2}{l+1}}\lambda_r^{\frac{1}{l+1}}} C_0.$$

For any other flow t which uses the link $s_{max}^{(r)}$ with $\lambda_t = \sum_{j:R_{jt}=1}\rho(l_j) \geq \rho(s_{max}^{(r)})$ the steady state throughput is given by:

$$x_t = \frac{1}{RTT_t^{\frac{2}{l+1}}\lambda_t^{\frac{1}{l+1}}} C_0.$$

Recall that we have defined the link $s_{max}^{(r)}$ as the most congested link on route of flow r in the sense of (1). This implies that $\lambda_r \leq M\rho(s_{max}^{(r)})$. Now

$$\frac{x_t}{x_r} = \left(\frac{RTT_r}{RTT_t}\right)^{\frac{2}{l+1}} \left(\frac{\lambda_r}{\lambda_t}\right) \leq \left(\frac{RTT_r}{RTT_t}\right)^{\frac{2}{l+1}} \left(\frac{M\rho(s_{max}^{(r)})}{\rho(s_{max}^{(r)})}\right)^{\frac{1}{l+1}} \leq$$

$$\leq \nu^{\frac{2}{l+1}} M^{\frac{1}{l+1}} < 1+\varepsilon$$

■

Remark: Note that for a single bottleneck topologies resource allocation given by $\frac{C}{RTT_i^{2/(l+1)}}$ is $((RTT_i^2), l+1)$ proportionally fair. Indeed, for any resource allocation (x_i), utility $U(x) = \sum_{i=1}^{N}\frac{RTT_i^2}{x_i^l}$, and link capacity c_0 we have(using Holder's inequality):

$$U(x)\cdot c_0 = \left(\sum_{i=1}^{N}\frac{RTT_i^2}{x_i^l}\right)\cdot(\sum_{i=1}^{N}x_i) =$$

$$\left(\sum_{i=1}^{N}\left(\frac{RTT_i^{2/(l+1)}}{x_i^{\frac{l}{l+1}}}\right)^{l+1}\right)^{\frac{1}{l+1}} \cdot \left(\sum_{i=1}^{N}\left(x_i^{\frac{l}{l+1}}\right)^{\frac{l+1}{l}}\right)^{\frac{l}{l+1}} \leq$$

$$\le \sum_{i=1}^{N} \frac{RTT_i^{2/(l+1)}}{x_i^{\frac{l}{l+1}}} \cdot x_i^{\frac{l}{l+1}} = \sum_{i=1}^{N} RTT_i^{2/(l+1)}$$

$U(x)$ is maximized if equality holds in the inequality above, which is equivalent to $x_i = \frac{C}{RTT_i^{2/(l+1)}}$ for some constant C.

It is interesting to notice that the fairness of well known eXplicit Congestion Control - XCP, features a similar property. Namely, in the recent paper [18] the authors proved that *"... given any network topology, one can choose a shuffling parameter, γ, sufficiently small so that the resulting allocation is close to max-min fairness. For any fixed $\gamma > 0$, however, there are topologies in which some flow rates can be far away from their max-min allocations"*. In the case of CL schemes for given network topology we can chose sufficiently large index l so that resulting allocations are close to max-min, but for any fixed l there are topologies in which some flow rates can be far away from their max-min allocations.

C. CC-l, a version of CHOKe-like scheme with additional field in IP header

As we noticed earlier, MLC becomes too computationally demanding when its index becomes large as it requires too many comparisons per packet for large number of user and large l. On the other hand if we use the basic idea from CSFQ [10] in which core routers have explicit information on current throughput for each flow, then the router can explicitly control ρ_0 and drop packets according to Definition 2.1. Recall that controlling h_M in MLC actually implicitly determines ρ_0 and essentially the effects of MLC and CC are same in terms of fairness.

III. Markov Active Yield (MAY) AQM's

In the previous section we have seen that the procedure of comparing an arriving packet with packets from the queue, and making drop decision based on this comparison, can lead to fairness that is arbitrarily close to max-min fairness. A basic problem with this strategy is that it can be computationally expensive to implement and it is therefore of interest to examine other methods of enforcing max-min fairness.

In order to enforce fairness among users with different levels of aggressiveness, and responsiveness, it is clear that more aggressive and less responsive users must be punished more. The question is "by how much" and how this strategy should be implemented in an efficient manner.

If an AQM does not differentiate among different flows, a packet drop will adversely affect less aggressive users. This is the source of unfairness among different TCP flows. The following synthetic example can be helpful in understanding the motivation for MAY.

Consider single bottleneck topology, where the congested link services N TCP users with round-trip times $RTT_1, \ldots, RTT_N$. For an AQM scheme Γ, denote by $\sigma(\Gamma)$ the set of serviced packets during the unit of time, and by $\delta(\Gamma)$ the set of dropped packets during the unit of time. If we denote by U_i and p_i the throughput and the loss rate of flow i, then the proportion σ_i of packets from flow i in $\sigma(\Gamma)$ is equal to $U_i \cdot a_1$ and the proportion δ_i of packets from flow i in $\delta(\Gamma)$ is equal to $U_i \cdot p_i \cdot a_2$, for some constants a_1 and a_2 independent of i. For the TCP flow i from the square root formula $U_i = \frac{C_0}{RTT_i \cdot \sqrt{p_i}}$ we conclude that

```
UpdateLossRates()
  t = 1;
  for $i = 1 : N$
     $\delta_i(t) = \frac{A'}{RTT_i}$;
  endfor;
  while forever
     $p_i(t) = \rho\delta_i(t)$;
     Drop packet from flow $i$ with prob. $p_i(t)$;
     Control $\rho$ such that utilization is $u_0$
     On the end of $t$-th time step do
        for i = 1:N
           $\delta_i(t+1) = \frac{NDrops_i(t)}{TotalDrops(t)}$;
        endfor;
        $t = t + 1$;
     enddo;
  endwhile;
```

Fig. 2. Positive feedback loop for calculating drop probabilities p_i.

$$\sigma_i \cdot \delta_i = \frac{A}{RTT_i^2}, \tag{2}$$

for some constant A that does not depend on i. For example if Γ is oblivious (like RED or BLUE), loss rate for each flow is constant ($p_i = p$) and $\sigma_i = \frac{A'}{RTT_i}$ and $\delta_i = \frac{A'}{RTT_i}$, for $A' = \frac{1}{\sum_{j=1}^{N} \frac{1}{RTT_j}}$. When Γ is $MLC(2)$, throughput of i-th flow (and therefore σ_i) is approximately proportional to $\frac{1}{RTT_i^{2/3}}$ and from (2) we conclude that $\delta_i = \frac{A''}{RTT_i^{4/3}}$ and $p_i = \frac{A'''}{RTT_i^{2/3}}$. Thus very aggressive TCP connections (ones with very small RTT) receive smaller throughput, than in oblivious case by increasing their loss rate, which gives additional throughput to connections with longer RTT (and smaller aggressiveness). Consider now the positive feedback system $(\delta(k))$ given in Figure 2: dropping probability for the flow i during the time step $t+1$ is proportional to the number of dropped packets during the time step t. If we start with vector $\delta(1) = A'(\frac{1}{RTT_1}, \ldots, \frac{1}{RTT_N})$, on the end of time step 1, $\delta(2)$ will be equal to $(\frac{B(1)}{RTT_i^{3/2}}, \ldots, \frac{B(1)}{RTT_N^{3/2}})$. Actually from the relation (2) we can deduce that $\delta(t) = (\frac{B(t)}{RTT_i^{e(t)}}, \ldots, \frac{B(1)}{RTT_N^{e(t)}})$ where sequence $e(t)$ satisfies $e(1) = 1$, and the following recurrence equation

$$e(t+1) = 1 + \frac{e(t)}{2}. \tag{3}$$

By taking $\hat{e}(t) = 2 - e(t)$, (3) is equivalent to $\hat{e}(t+1) = \frac{\hat{e}(t)}{2}$. Thus $e(t) = 2 - 2^{-(t-1)}$, and from (2), $U_i(t) = \frac{C(t)}{(RTT_i)^{2^{-(t-1)}}}$ does depend on RTT_i for for large t, meaning that asymptotically all flows achieve same throughput.

Now we proceed with description of MAY. Our basic idea is to keep information of recently dropped packets and to use that statistics to regulate the drop probability for each of the flows.

Basic data structure used is a hash table H that stores quadruples $(FlowID, p(FlowID), TS(FlowID), ND(FlowID))$, where $p(FlowID)$ is the drop probability of the flow given

u_0	desired utilization
Δ	length of update period
$PctNew$	percentage of new drop entries in H
q_w	weighted average parameter
T_0	Timeout value

TABLE I
PARAMETERS OF MAY.

```
1  OnPacketArrival($pkt, FlowID$)
2    if $FlowID \in H$
3      with probability $p(FlowID)$ do
4        $drop(pkt)$;
5        $ND(FlowID) + +$;
6        $TS(FlowID) = now$;
7      enddo
8    else with probability $q_0$ do
9      $create(FlowID, H)$
10     $p(FlowID) = 0$;
11     $ND(FlowID) = 1$;
12     $TS(FlowID) = now$;
13     $drop(pkt)$;
14   enddo

15 Update($H$)
16   if $now - LastUpdate > \Delta$
17     $\nu_{old} = \nu$;
18     $\nu = \nu + (utilization - u_0)$;
19     $TotalND = \sum_{FID \in H} ND(FID)$;
20     for all $FID$ in $H$
21       if $now - TS(FID) > T_0$
22         $remuve(FID, H)$
23       else
24         $p(FID) = \nu((1 - q_w)\frac{p(FID)}{\nu_{old}} + q_w \frac{ND(FID)}{TotalND})$;
25         $ND(FlowID) = 0$;
26       endelse
27     endfor
28     $q_0 = PctNew \cdot \frac{TotalND}{TotalNArrivals}$;
29     $LastUpdate = now$;
30   endif
```

Fig. 3. Pseudocode of MAY.

by identifier $FlowID$, $TS(FlowID)$ is the time stamp that tracks the time of last update of the given hash table entry and $ND(FlowID)$ number of dropped packets from the flow $FlowID$ during the current update interval of length Δ. On each packet arrival its $FlowID$ is calculated. If there exist entry in H that corresponds to $FlowID$, the arriving packet is dropped with probability $p(FlowID)$, we call these drops *old drops*. In order to accept new flows to H, we allow $PctNew\%$ of drops to be reserved to for flows that does-not have entry in H. The drop probability of flow FID is product of control variable ν and weighted average of proportion of drops corresponding to FID (see line 24). Finally a control variable ν determines size of drop probabilities and therefore the utilization: if current utilization is less than desired (u_0) ν should be decreased to allow lower drop probabilities and increase utilization, while if current utilization is greater than u_0, then ν should be increased to allow higher drop probabilities and decrease utilization.

A. Analysis of MAY

Having described the main algorithm, we will now prove that under following assumptions, arbitrary AIMD flows (flows with arbitrary linear increase parameter and arbitrary multiplicative decrease parameter), sharing a single link asymptotically obtain equal amount of available throughput. Throughout this section we make the following assumptions.

Assumption 3.1: There are N long-lived flows that use a congested link and all of them employ AIMD congestion control algorithms with an additive increase parameter $\gamma_i > 0$ packets per unit of time, and a multiplicative decrease $\delta_i \in (0,1)$, $i = 1, 2, \ldots, N$.

Assumption 3.2: The network is in steady state: the utilization is equal to u_0 and the drop probability for i-th flow $p(i)$ is constant and equals $\lambda_i \cdot \nu > 0$.

The first assumption defines the type of congestion control algorithms employed by users on the link. The second assumption is actually saying that in the interval of interest the drop probabilities do not change. Note that under static network conditions, due to of statistical multiplexing, the proportion of drops from certain flow in the drop history is asymptotically constant. In our model we assume that it is constant at each instant of time.

Preamble to main result :

In order to precisely formulate the main result, we present a model of the evolution of throughput for the i'th flow traversing the link.

Let $U_k^{(i)}$ be the throughput of flow (i) measured after k units of time. Since we have assumed that MAY is in steady state, the loss probability for a packet from flow (i) is constant and is equal to $\mu_i = \lambda_i \nu$. Having this, we can consider $U_k^{(i)}$ as a Markov chain on R^+ given by the transition:

$$U_{k+1}^{(i)} = U_k^{(i)} + \gamma_i \quad \text{with probability} \quad e^{-\lambda_i \nu U_k^{(i)}}$$

$$U_{k+1}^{(i)} = \delta_i U_k^{(i)} \quad \text{with probability} \quad 1 - e^{-\lambda_i \nu U_k^{(i)}}.$$

We can scale the Markov chain $U_k^{(i)}$, such that additive increase is equal to 1: $W_k^{(i)} = (1/\gamma_i) U_k^{(i)}$.

$$W_{k+1}^{(i)} = W_k^{(i)} + 1 \quad \text{with probability} \quad e^{-\theta_i W_k^{(i)}}$$

$$W_{k+1}^{(i)} = \delta_i W_k^{(i)} \quad \text{with probability} \quad 1 - e^{-\theta_i W_k^{(i)}}.$$

Here $\theta_i = \gamma_i \lambda_i \nu$. In [20], techniques for the analysis of Markov the chain $W_k^{(i)}$ were developed. Employing these tools we will prove the main result of this section:

Theorem 3.1: Under assumptions (1-3), the asymptotic throughput of flow (i)

$$T^{(i)}(\gamma_i, \delta_i) = \lim_{m \to \infty} \frac{1}{m} \sum_{k=1}^{m} U_k^{(i)}$$

converges almost surely, and for small ν, $T^{(i)}(\gamma_i, \delta_i)$ does not depend on the additive increase parameter γ_i or the multiplicative decrease parameter δ_i. Formally, there exist a constant $c > 0$ such that:

$$T^{(i)}(\gamma_i, \delta_i) = \frac{1}{c\nu + o(\sqrt{\nu})} + o(\frac{1}{\sqrt{\nu}}) \tag{4}$$

Proof: Before we begin the proof, we introduce some notation from [20]. By $V_m^{(i)}$ we denote the Markov process with states of $W_k^{(i)}$ just after packet losses, i.e.: $V_m^{(i)} = W_{s_m}^{(i)}$, where s_m is the m-th smallest k such that $W_k^{(i)} = \delta_i W_{k-1}^{(i)}$. It is clear that $V_k^{(i)}$ is Markov chain with transition given by:

$$V_{k+1}^{(i)} = \delta_i (V_k^{(i)} + G_{V_k^{(i)}}^{(i)})$$

where $G_x^{(i)}$ is random variable on positive integers defined by

$$P(G_x^{(i)} \geq m) = \prod_{k=x}^{x+m-1} e^{-\theta_i k}.$$

The following proposition is Corollary 1 from [20].

Lemma 3.1: If the initial state of the Markov chain $V_k^{(i)}$ is 0, then as θ_i goes to 0, the Markov chain $\sqrt{\theta_i} V_k^{(i)}$ converges in distribution to the Markov chain $\overline{V}_k$ defined by $\overline{V}_0 = 0$ with transition:

$$\overline{V}_{k+1} = \delta_i (\overline{V}_k + \overline{G}_{\overline{V}_k}) \tag{5}$$

where $\overline{G}_x$ is a random variable given by the distribution $P(G_x \geq y) = e^{-\frac{1}{2}y^2 - xy}$.

From the Proposition 7 [20] we conclude that the Markov chain with transition given by (5), has a unique stationary distribution $\overline{V}_\infty$. From Proposition 5[20] we get that

$$E(\overline{V}_\infty) = \frac{\delta_i}{1-\delta_i} E(G_{\overline{V}_\infty}). \tag{6}$$

Further, from Theorem 1[20] and Lemma (3.1), we conclude that as θ_i goes to 0, the invariant distribution of $\sqrt{\theta_i} V_m^{(i)}$ converges to $\overline{V}_\infty$ and therefore

$$E(G_{V_\infty^{(i)}}) = \frac{1}{\sqrt{\theta_i}} E(G_{\overline{V}_\infty}) + o(\frac{1}{\sqrt{\theta_i}}) \tag{7}$$

On the other hand $G_{V_\infty^{(i)}}$ determines the times between drops from the flow (i). Thus the asymptotic proportion of drops from the flow (i) in the drop history is determined by frequency of drops and therefore equal to

$$\lambda_i = \frac{c}{E(G_{V_\infty^{(i)}})} \tag{8}$$

for some constant c that is equal to average time between two consecutive drops at MAY link.

Now we are ready to conclude the proof of the Theorem. From the Proposition 9 from [20] we have almost sure convergence of asymptotic throughput to $T^{(i)}(\gamma_i, \delta_i)$ and:

$$T^{(i)}(\gamma_i, \delta_i) = \gamma_i \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} U_k^{(i)}$$

$$= \frac{\gamma_i}{\sqrt{\theta_i}} \frac{\delta_i}{(1-\delta_i)E(\overline{V}_\infty)} + o(\frac{1}{\sqrt{\nu}}). \tag{9}$$

Combining (9) with (6),(7) and (8) we obtain:

$$T^{(i)}(\gamma_i, \delta_i) = \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \nu}} \frac{\delta_i}{(1-\delta_i)\frac{\delta_i}{1-\delta_i}E(G_{\overline{V}_\infty})} + o(\frac{1}{\sqrt{\nu}})$$

$$= \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \nu}} \frac{1}{E(G_{\overline{V}_\infty})} + o(\frac{1}{\sqrt{\nu}})$$

$$= \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \nu}} \frac{1}{\sqrt{\gamma_i \lambda_i \nu} E(G_{V_\infty^{(i)}}) + o(1)} + o(\frac{1}{\sqrt{\nu}})$$

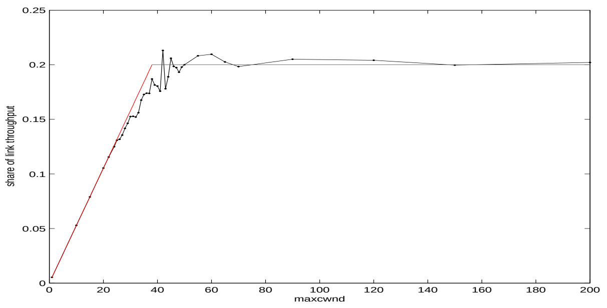

Fig. 4. Maximal window size vs share of throughput

$$= \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \nu}} \frac{1}{\sqrt{\gamma_i \lambda_i \nu}\frac{c}{\lambda_i} + o(1)} + o(\frac{1}{\sqrt{\nu}}).$$

Thus we have

$$T^{(i)}(\gamma_i, \delta_i) = \frac{1}{c\nu + o(\sqrt{\nu})} + o(\frac{1}{\sqrt{\nu}})$$

and the proof is finished. ■

B. Discussion of MAY

Fairness in general network topologies. We have observed empirically that the bandwidth allocation of AIMD users over topologies with multiple congested links operating MAY schemes is very close to being max-min fair. To compare how far bandwidth allocations $U = (U_1, \ldots, U_N)$ deviate from max-min fair bandwidth allocations, $U_{mm} = (U_{1,mm}, \ldots, U_{N,mm})$, one may use Jain's index the given by:

$$j(U) = \frac{\left(\sum_{i=1}^{N} \frac{U_i}{U_{i,mm}}\right)^2}{N \sum_{i=1}^{N} \left(\frac{U_i}{U_{i,mm}}\right)^2}. \tag{10}$$

Clearly, $j(U)$ has a global maximum 1 that is attained at $U = U_{mm}$ and since it is continuous, by measuring how far the index is from 1, one can get some intuition on how far is vector U from U_{mm}.

In the Section IV-B the following indices for DropTail, RED and MAY were obtained:

$$j(U_{DropTail}) = 0.35, \; j(U_{RED}) = 0.73, \; j(U_{MAY}) = 0.985.$$

We believe that such fairness properties of MAY follows from the fact that the drop history at a congested link is made up mainly from packets from flows which get most bandwidth and therefore flows which are not bottlenecked at the link (with average rate less then maximal average rate at the link) experience very few drops at that link. This means that a large majority of packets dropped at the link are from the flows bottlenecked at that link. If we ignore all other flows in network, we are in a situation where all bottlenecked flows represent the single bottleneck case, and therefore results from the previous subsection apply.

To illustrate the fact that flows with average rate less than fair share are afforded extra protection at a link employing MAY we performed the following experiment using the network simulator $ns-2$. We ran a simulation with 5 TCP flows: $f1, f2, f3, f4$ and $f5$ over a single congested link employing MAY. Flows $f1, f2, f3$ and $f4$ have an unlimited maximal window size ($maxcwnd$), while $maxcwnd$ for flow $f5$ is varied from 1 to 200 packets. For a maximal window size 38, flow $f5$ would get approximately its max-min fair share of available bandwidth. As we can see from Figure 4, the throughput obtained by flow $f5$ in the presence of MAY drops for $maxcwnd$ from [1,38] is almost equal to throughput which would be achieved without any drops (straight line). For $maxcwnd \in [38, 200]$, MAY keeps the bandwidth allocated to flow $f5$ close its fair share of 20%.

Isolating the high-rate nonresponsive flow. Nonresponsive flows that have sending rate higher than responsive flows bottlenecked at given link have significantly higher drop rates and can be easily identified, but we do not discuss it here because of space limitation.

Parameter calibration. The initial tests show that MAY is highly robust to the choice of parameters. The update interval Δ should be taken to cover several "typical" RTT, thus to belong to interval $[500, 5000]ms$. PctNew should be small positive number (for example taken in range $[0.01, 0.1]$) to allow new flows to get an H entry. The weighted average q_w should be chosen to allow averaging over several update interval: $q_w \in [0.01, 0.1]$. Timeout T_0 depends on the available memory and can be controlled based on memory consumption: the higher the memory consumption the lower T_0 should be. Self tuning of other parameters is also possible but is out of scope of the present paper.

IV. Experimental results

In this section we briefly describe some experiments that demonstrate the behavior of proposed AQM schemes.

A. Single bottleneck

The first set of experiments are designed to demonstrate the fairness properties of the proposed AQM schemes over single link. Specifically, we present results for a single link with service rate of $80Mbps$ that services 100 long-lived TCP users with round trip times uniformly distributed in range $40 - 440ms$ and approximately $10ms$ of queueing delay. To provide baseline results, we include the performance of RED for the same scenario. Share of total throughput taken by each of 100 flows in these 3 schemes is depicted in Figure 5

It can be clearly seen from our results that that the fairness of RED is approximately proportional to the inverse of RTT. This is in accordance with our theoretical results under assumption that drop probability flow independent and is also consistent with results and observations made by other authors [28], [29]. It can also be observed that the fairness of MLC with index 2 is proportional to $1/RTT^{2/3}$ as predicted by Theorem 2.1. The MLC parameters used are in the experiment are: $index = 2$, $SamplingTime = 100ms$, $\gamma = 0.99$, $\delta = 1.01$, $u_0 = 0.98$. The MAY parameters used in the experiment are: $\Delta = 500ms$, $u_0 = 0.98$, $PctNew = 0.05$, $q_w = 0.05$, $T_0 = 10sec$.

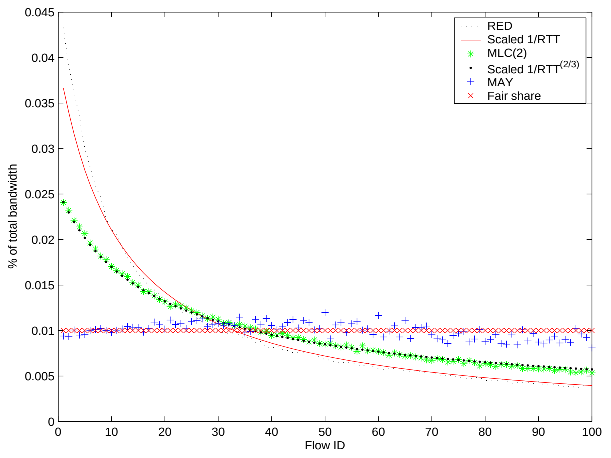

Fig. 5. Scaled throughput for 100 flows over congested link employing RED, MLC(2) and MAY.

Fig. 6. Network topology

B. Multiple bottleneck topologies

Our second set of simulations demonstrate Theorem 2.2. The network topology that we considered is given in Figure 6. Here, we consider a network of 24 nodes: $n1 - n5, m1 - m5, p1 - p5, q1 - q5$, and $c1$, $c2$, $c3$, $c4$ and 30 flows traversing the network as follows: $n(i) \rightarrow p(i); n(i) \rightarrow q(i), m(i) \rightarrow p(i); m(i) \rightarrow q(i); n(i) \rightarrow m(i); p(i) \rightarrow q(i)$ where $i = 1, 2, 3, 4, 5$.

The delays on each of the links in ms are defined as follows:

$$\begin{aligned} ni \rightarrow c1 &: 40 * i + 1 \\ pi \rightarrow c3 &: 40 * i + 1 \\ mi \rightarrow c2 &: 40 * i + 1 \\ qi \rightarrow c4 &: 40 * i + 1 \end{aligned}$$

and the delays $c1 - c2$, $c2 - c3$, $c3 - c4$ are $10ms$. The capacities of all links are $10Mbps$. With this topology, the max min fair shares are 0.5Mbps for 20 flows that uses link $c2 - c3$, and $1Mbps$ for other 10 flows ($n(i) \rightarrow m(i)$ and $p(i) \rightarrow q(i)$).

Each flow uses the standard TCP-SACK algorithm, with a packet size 1000B. The aggressiveness of each flow is mainly determined by its RTT. The behavior of the network is evaluated with each link $c1 - c2$, $c2 - c3$ and $c3 - c4$ using: DropTail, RED, CC-1, CC-2, CC-4, CC-5, CC-9, CC-17 and MAY with a queue size 100 packets. CC-l parameters are: $index = l$, $SamplingTime = 100ms$, $\gamma = 0.99$, $\delta = 1.01$, $u_0 = 0.98$. The MAY parameters used in the experiment are: $\Delta = 500ms$, $u_0 = 0.98$, $PctNew = 0.05$, $q_w = 0.1$, $T_0 = 10sec$.

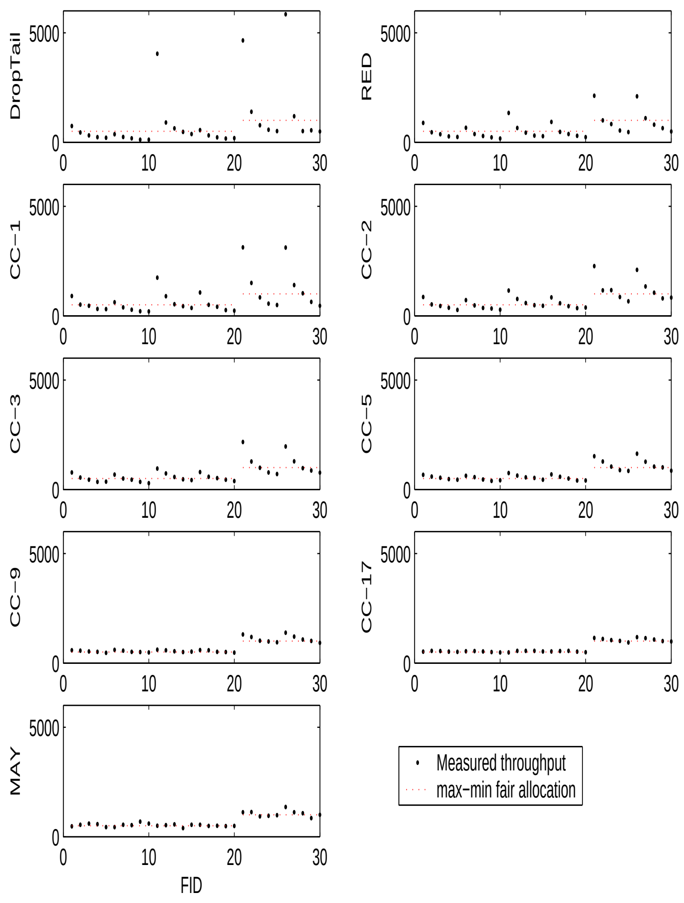

Fig. 7. Amount of bandwidth taken by each of 30 flows. x-axis is flowID, y-axis is of time aberrated throughput in Kbps. Droptail, RED, CL-1, CL-2, CL-3, CL-5, CL-9, CL-17, MAY

Figure 7 depicts 9 scenarios, one for each dropping scheme. Each plot depicts the amount of throughput taken by each of 30 flows. The doted line represents the max-min fair share of bandwidth while points represent measured share of bandwidth for each of 30 flows in the network. We can see significant unfairness in DropTail, RED and CC-1 cases. As we increase index of CL scheme, we obtain share of bandwidth very close to max-min share as expected (Theorem 2.2). Jain's indices, defined by (10) for each of these AQM's are:

$$j(U_{DTail}) = 0.345,\ j(U_{RED}) = 0.731,\ j(U_{CC-1}) = 0.661$$

$$j(U_{CC-2}) = 0.846,\ j(U_{CC-3}) = 0.884,\ j(U_{CC-5}) = 0.956$$

$$j(U_{CC-9}) = 0.989,\ j(U_{CC-17}) = 0.997,\ j(U_{MAY}) = 0.985$$

V. Summary

In this paper we developed two AQM schemes for enforcing max-min fairness in TCP networks: MLC(l) and MAY. MLC(l) is a stateless scheme and belongs to class of queue management schemes that we call CHOKe-like. We showed analytically that by increasing index l, the resource allocation among TCP users using network of MLC(l) queues converge to max-min fair. To the best of our knowledge no stateless AQM scheme for enforcing max-min fairness exists in the literature. However, the price for being close to max-min fairness must be payed by large number of comparisons.

Our second scheme, MAY, exploits paradigm in which the drop decision is based solely on short history of previously dropped packets. It turns out that this information is enough to enforce fairness among various levels of aggressiveness and responsiveness (modelled as AIMD). From practical point of view, keeping information of only dropped packets can significantly reduce the memory requirements for storing the state.

References

[1] S. Athuraliya, D. Lapsley, S. Low. "Enhanced Random Early Marking algorithm for Internet flow control". Proceedings of IEEE INFOCOM, Tel Aviv, Israel, 2000.

[2] V. Jacobson. "Congestion avoidance and control". Proceedings of SIGCOMM, Vancouver, BC, Canada, 1988.

[3] S. Floyd, V. Jacobson. "Random early detection gateways for congestion avoidance". IEEE/ACM Transactions on Networking, vol. 1, pp. 397 - 413, Aug. 1993.

[4] W. Feng, K.G. Shin, D.D. Kandlur, D. Saha. "The BLUE active queue management algorithms". *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, 513-528, August 2002.

[5] C. Hollot, V. Misra, D. Towsley, W.B. Gong. "Analysis and design of controllers for AQM routers supporting TCP flows". *IEEE Transactions on Automatic Control*, pp. 945-959 June, 2002.

[6] S. Kunniyur, R. Srikant. "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management". *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, 286-299, April 2004.

[7] A. Demers, S. Keshav, S. Shenker. "Analysis and simulation of a fair queueing algorithm". Proceedings of ACM SIGCOMM, Austin, TX, 1989.

[8] M. Shreedhar, G. Varghese. "Efficient fair queueing using deficit round-robin". *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, June 1996.

[9] D. Lin, R. Morris. "Dynamics of random early detection". Proceedings of ACM SIGCOMM, Cannes, France, 1997.

[10] I. Stoica, S. Shenker, H. Zhang. "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks". *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, 33-46, February 2003.

[11] R. Pan, L. Breslau, B. Prabhakar, S. Shenker. "Approximate Fairness through Differential Dropping". ACM SIGCOMM Computer Communication Review Volume 33 , Issue 2, April 2003

[12] R. Mahajan, S. Floyd, D. Wetherall. "Controlling high-bandwidth flows at the congested router". Proceedings of IEEE ICNP, Riverside, CA, USA, 2001.

[13] D. Katabi, M. Handley, C. Rohr. "Internet Congestion Control for Future High Bandwidth-Delay Product Environments". Proceedings of ACM SIGCOMM, Pittsburgh, PA, USA, 2002.

[14] B. Wydrowski, M. Zukerman. "MaxNet: A congestion control architecture for maxmin fairness". IEEE Communications Letters, vol. 6, no. 11, Nov. 2002, pp.512-514.

[15] Y. Yang, S. Lam. "General AIMD Congestion Control". Proceedings ICNP 2000, Osaka, Japan, 2000.

[16] S. Floyd. "HighSpeed TCP for Large Congestion Windows". RFC 3649, Experimental, December 2003.

[17] T. Kelly. "Scalable TCP: Improving Performance in Highspeed Wide Area Networks". ACM SIGCOMM Computer Communication Review Volume 33, Issue 2, April 2003.

[18] S. Low, L. Andrew, B. Wydrowski. "Understanding XCP: Equilibrium and Fairness". Proceedings of IEEE Infocom, Miami, FL, March 2005.

[19] T. V. Lakshman, U. Madhow. "The performance of TCP/IP for networks with high bandwidth-delay products and random loss". *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, 336-350, June 1997.

[20] V. Dumas, F. Guillemin, P. Robert. "A Markovian analysis of additive-increase multiplicative-decrease algorithms". Adv. in Appl. Probab. 34 (2002), no. 1, 85111.

[21] L. Breiman. "Probability". SIAM classics in Applied Math. 1992.

[22] M. Kodialam, T. V. Lakshman, Shantidev Mohanty. "Run based Traffic Estimator (RATE): A Simple, Memory Efficient Scheme for Per-Flow Rate Estimation". Proceedings of IEEE INFOCOM, Hong Kong, March 2004.

[23] Mo. J, Walrand. J. “Fair end-to-end window-based congestion control”. IEEE/ACM Transactions on Networking, Vol. 8, No. 5 October, 2000.
[24] C. Estan, G. Varghese. “New Directions in Traffic Measurement and Accounting”. SIGCOMM, August 2002
[25] R. Srikant. “The Mathematics of Internet Congestion Control”. Birkhauser, 2004.
[26] K. Fall, S. Floyd. “Router mechanisms to support end-to-end congestion control”. [online] ftp://ftp.ee.lbl.gov/papers/collapse.ps.
[27] R. Pan, B. Prabhakar, K. Psounis. “CHOKe: A stateless AQM scheme for approximating fair bandwidth allocation”. Proceedings of IEEE INFOCOM, Tel Aviv, Israel, 2000.
[28] A. Abouzeid, S. Roy. “Analytic understanding of RED gateways with. multiple competing TCP flows”. Proceedings of IEEE GLOBECOM, 2000.
[29] S. Floyd. “Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: Oneway Traffic”. ACM Computer Communication Review, 30 - 47, Vol. 21 , Issue 5, October 1991.
[30] A. Das, D. Dutta, A. Goel, A. Helmy, J. Heidemann. “Low State Fairness: Lower Bounds and Practical Enforcement”. Proceedings of the IEEE INFOCOM, Miami, FL, USA, March 2005.
[31] T. J. Ott, T. V. Lakshman, L. H. Wong. “SRED: Stabilized RED”. Proceedings IEEE INFOCOM, New York, March 1999.
[32] B. Suter, T.V. Lakshman, D. Stiliadis, A.K. Choudhury. “Buffer management schemes for supporting TCP in gigabit routers with per-flow queueing”. IEEE Journal on Selected Areas of Communications 17 (6) (1999) 1159-1170.

APPENDIX

A. Proof of Theorem 2.1

Let s_k be the time of k-th loss, ie: s_k is the k-th smallest m which satisfies $U_m^{(\rho)} = \frac{1}{2}U_{m-1}^{(\rho)}$. Define $Z_0^{(\rho)} = 0$ and $Z_k^{(\rho)} = U_{s_k}^{(\rho)}$. It follows that $Z_k^{(\rho)}$ is also a Markov chain and the following theorem provides insight into the behavior of $Z_k^{(\rho)}$ for small values of ρ.

Proposition 1.1: There exist a Markov chain $\overline{V}_k$, such that for $\rho > 0$,

$$Z_k^{(\rho)} = \frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}}\overline{V}_k + \frac{1}{\rho^{\frac{1}{l+1}}}R_k^{(\rho)},$$

where the stochastic process $R_k^{(\rho)}$ converge to zero in distribution as $\rho \to 0$.

Preamble to Proof of Proposition 1.1

The Markov chain $U_k^{(\rho)}$ can be written as $U_k^{(\rho)} = RTT^2 W_k^{(\alpha)}$, where $\alpha = \rho/RTT^{2l}$ and

$$W_{k+1}^{(\alpha)} = W_k^{(\alpha)} + 1 \quad \text{with probability} \quad e^{-\alpha(W_k^{(\alpha)})^l}$$
$$W_{k+1}^{(\alpha)} = \frac{1}{2}W_k^{(\alpha)} \quad \text{with probability} \quad 1 - e^{-\alpha(W_k^{(\alpha)})^l}.$$

Analogously, we can define another associated Markov chain with states of $W_k^{(\alpha)}$ just after congestion: $V_k^{(\alpha)} = Z_k^{(\rho)}/RTT^2$. Its transition is given by

$$V_{k+1}^{(\alpha)} = \frac{1}{2}(V_k^{(\alpha)} + G_{V_k^{(\alpha)}}^{(\alpha)}).$$

Here $G_n^{(\alpha)}$ is random variable determined by distribution:

$$P(G_x^{(\alpha)} \geq y) = \prod_{k=x}^{x+\lfloor y \rfloor} e^{-\alpha k^l}.$$

Lemma 1.1: Let $x \geq 0$. Then as α goes to 0, the random variable $\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)}$ converges in distribution to a random variable $\overline{G}_x$ such that for $y \geq 0$

$$P(\overline{G}_x \geq y) = e^{-\frac{1}{l+1}((x+y)^{l+1} - x^{l+1})}. \tag{11}$$

Proof: Let $x, y \geq 0$ and $\alpha < 1$. Then

$$P(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y) = P(G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y/\alpha^{\frac{1}{l+1}})$$
$$= \prod_{k=x/\alpha^{\frac{1}{l+1}}}^{\lfloor (x+y)/\alpha^{\frac{1}{l+1}} \rfloor} e^{-\alpha k^l}.$$

Taking logarithms we conclude:

$$ln(P(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y)) = -\alpha \sum_{k=x/\alpha^{\frac{1}{l+1}}}^{\lfloor (x+y)/\alpha^{\frac{1}{l+1}} \rfloor} k^l \longrightarrow$$
$$\longrightarrow -\alpha \int_{x/\alpha^{\frac{1}{l+1}}}^{(x+y)/\alpha^{\frac{1}{l+1}}} t^l dt = \frac{1}{l+1}((x+y)^{l+1} - x^{l+1}).$$

■

Definition 1.1: The sequence $\overline{V}_n$ denotes a Markov chain given by $\overline{V}_0 = 0$ and transitions:

$$\overline{V}_{n+1} = \frac{1}{2}(\overline{V}_n + \overline{G}_{\overline{V}_n}).$$

Here $\{\overline{G}_x : x \geq 0\}$ is a family of random variables independent of $\overline{V}_n$ such that the distribution of $\overline{G}_x$ is given by (11).

Lemma 1.2: The Markov chain $\alpha^{1/(l+1)} V_n^{(\alpha)}$ converges in distribution to the Markov chain $\overline{V}_n$.

Proof: The proof of this fact follows same lines as proof of Corollary 1 in paper [20]. Namely, using uniform continuity of the mapping $t \to t^l$, and techniques from the proof of Proposition 1 from [20], it follows that for any $K > 1$

$$\lim_{\alpha \to 0} \sup_{\alpha^{1/(l+1)} < x, y < K} |P(\overline{G}_x \geq y) - P(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y)| = 0.$$

Then, using the previously established uniform convergence, it follows that for any continuous function f on R^+ with compact support:

$$\lim_{\alpha \to 0} \sup_{x > \alpha^{1/(l+1)}} |E(f(\overline{G}_x)) - E(f(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)}))| = 0.$$

Finally, again following the same lines as in proof of Proposition 2 from [20] we can conclude desired convergence in distribution. ■

The previous lemma allows us to approximate the Markov chain $V_k^{(\alpha)}$ with $\overline{V}_k$ with initial value $\overline{V}_0 = \alpha^{1/(l+1)} V_0^{(\alpha)} = 0$. We are ready to give:

Proof of Proposition 1.1. Recall that $Z_k^{(\rho)}$ represents throughput just after packet loss and that $\alpha = \rho/RTT^{2l}$. Then for small ρ we can write[5]

$$Z_k^{(\rho)} = \frac{1}{RTT^2} V_k^{(\alpha)} = \frac{1}{RTT^2}(\frac{1}{\alpha^{1/(l+1)}}\overline{V}_k + o(\frac{1}{\alpha^{\frac{1}{l+1}}})) =$$
$$= \frac{RTT^{2l/(l+1)}}{RTT^2} \frac{1}{\rho^{1/(l+1)}} \overline{V}_k + o(\frac{1}{\rho^{\frac{1}{l+1}}}) =$$

[5]Here, $o(\frac{1}{\alpha^{\frac{1}{l+1}}})$ represent random variable $R(\alpha)$ such that $R(\alpha)/\alpha^{\frac{1}{l+1}} \to 0$ as $\alpha \to 0$

$$= \frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}}\overline{V}_k + o(\frac{1}{\rho^{\frac{1}{l+1}}})$$

■

Thus the Markov chain $Z_k^{(\rho)}$ of throughput after congestion events can be approximated with the Markov chain $\frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}}\overline{V}_k$ for small values of ρ as was claimed.

At this point we are concentrating on $\overline{V}_k$ and following the basic ideas from [20]. We shall prove that $(\overline{V}_k)^{l+1}$ is autoregressive(AR) process with unique invariant distribution that can be explicitly written.

Theorem 1.1: For the Markov chain $\overline{V}_k$, the process $\overline{V}_k^{l+1}$ is autoregressive with following representation:

$$\overline{V}_{k+1}^{l+1} = \frac{1}{2^{l+1}}(\overline{V}_k^{l+1} - (l+1)E_n)$$

where E_n is an IID sequence of exponentially distributed random variables with parameter 1: $P(E_n \geq y) = e^{-y}$. Moreover the Markov chain $\overline{V}_k$ has unique invariant distribution represented by the random variable $\overline{V}_\infty$ which satisfy:

$$\overline{V}_\infty = \sqrt[l+1]{(l+1)\sum_{n=1}^{\infty}\frac{1}{2^{n(l+1)}}E_n}, \qquad (12)$$

and the process defined with this invariant distribution as initial distribution for $\overline{V}_0$ is ergodic.

Proof: By definition we have

$$\overline{V}_{n+1} = \frac{1}{2}(\overline{V}_n + \overline{G}_{\overline{V}_n}).$$

Let $E_n = \frac{1}{l+1}(2^{l+1}\overline{V}_{n+1}^{l+1} - \overline{V}_n^{l+1})$. Then:

$$P(E_n \geq y) = P(\frac{1}{l+1}(2^{l+1}\overline{V}_{n+1}^{l+1} - \overline{V}_n^{l+1}) \geq y) =$$

$$= P((\overline{V}_n + \overline{G}_{\overline{V}_n})^{l+1} \geq (l+1)y + \overline{V}_n^{l+1})$$

$$= P(\overline{G}_{\overline{V}_n} \geq ((l+1)y + \overline{V}_n^{l+1})^{1/(l+1)} - \overline{V}_n) =$$

$$= e^{-\frac{1}{l+1}((l+1)y + \overline{V}_n^{l+1} - \overline{V}_n^{l+1})} = e^{-y}$$

Thus, the first part of the Theorem is proved. To prove the second part notice that random variable given by (12) represents an invariant probability. To prove its uniqueness we use Theorem 7.16 from [21]. It is enough to prove that there are no two disjoint sets $A_1, A_2 \subset R^+$ such that for $i = 1, 2$ and all $x \in A_i$, $P(\overline{V}_2 \in A_i | \ \overline{V}_1 = x) = 1$. Suppose that there exist two such sets A_1, A_2. Let $x_1 \in A_1$. Then since G_{x_1} has positive density we conclude that $[\frac{x_1}{2}, \infty) \setminus A_1$ has Lebesgue-measure 0. Similarly for $x_2 \in A_2$, $[\frac{x_2}{2}, \infty) \setminus A_2$ has Lebesgue-measure 0 which means that A_1 and A_2 cannot be disjoint. ■

Now we are ready to prove that for small values of ρ_0, the steady state throughput can be approximated by $\frac{1}{RTT^{\frac{2}{l+1}}\rho_0^{\frac{1}{l+1}}}D_{CL}$, where D_{CL} does not depend on ρ_0 nor l.

Proof: (of the Theorem 2.1)If we write $U_i^{(\rho)} = RTT^2 W_k^{(\alpha)}$, where $\alpha = \rho/RTT^{2l}$, the Theorem is an immediate consequence of the Theorem 1.1 and Proposition 9 from [20] for a multiplicative decrease factor $\delta = \frac{1}{2}$.[6] ■

[6]Here constant $\frac{3}{4} = \frac{1+\delta}{2\delta(1-\delta)}$, for $\delta = 0.5$.